# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report

from sklearn.model\_selection import train\_test\_split

import joblib

# Set random seed for reproducibility

np.random.seed(42)

# Step 1: Create a synthetic dataset (for demonstration)

# Features: Time (seconds since first transaction), Amount, V1-V4 (anonymized features), Class (0: Non-Fraud, 1: Fraud)

data = {

'Time': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],

'Amount': [100.0, 50.0, 2000.0, 75.0, 150.0, 3000.0, 25.0, 500.0, 10000.0, 80.0],

'V1': [-1.359, 1.191, -5.0, 0.966, -0.185, -7.0, 1.792, -0.418, -10.0, 1.257],

'V2': [0.072, -0.173, 4.0, -0.287, 0.669, 5.5, -0.863, 0.403, 7.0, -0.211],

'V3': [2.536, 0.405, -6.0, 1.798, 1.974, -8.0, 0.095, 0.762, -12.0, 0.988],

'V4': [1.378, -0.338, 3.5, -0.094, 0.456, 4.0, -0.631, 0.175, 6.0, -0.403],

'Class': [0, 0, 1, 0, 0, 1, 0, 0, 1, 0] # 0: Non-Fraud, 1: Fraud

}

df = pd.DataFrame(data)

# Step 2: Data Preprocessing

# Separate features and target

X = df.drop('Class', axis=1)

y = df['Class']

# Scale Time and Amount

scaler = StandardScaler()

X[['Time', 'Amount']] = scaler.fit\_transform(X[['Time', 'Amount']])

# Step 3: Train a Random Forest Classifier

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train the model

rf\_model = RandomForestClassifier(n\_estimators=50, random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Evaluate the model

y\_pred = rf\_model.predict(X\_test)

print("\nModel Performance on Test Set:")

print(classification\_report(y\_test, y\_pred))

# Save the model and scaler

joblib.dump(rf\_model, 'fraud\_detection\_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

# Step 4: Simulate Real-Time Transaction Processing

# Example new transactions (mimicking incoming data)

new\_transactions = pd.DataFrame({

'Time': [10, 11, 12],

'Amount': [120.0, 4500.0, 60.0],

'V1': [1.0, -8.0, 0.8],

'V2': [-0.2, 6.0, -0.3],

'V3': [1.5, -10.0, 1.2],

'V4': [-0.5, 5.0, -0.1]

})

# Load the model and scaler

model = joblib.load('fraud\_detection\_model.pkl')

scaler = joblib.load('scaler.pkl')

# Scale the new transactions

new\_transactions[['Time', 'Amount']] = scaler.transform(new\_transactions[['Time', 'Amount']])

# Predict fraud

predictions = model.predict(new\_transactions)

probabilities = model.predict\_proba(new\_transactions)[:, 1] # Probability of fraud

# Step 5: Output Results

print("\nNew Transaction Analysis:")

for i, (pred, prob) in enumerate(zip(predictions, probabilities)):

transaction = new\_transactions.iloc[i]

status = "Fraud" if pred == 1 else "Non-Fraud"

print(f"\nTransaction {i+1}:")

print(f"Time: {transaction['Time']:.2f}, Amount: ${transaction['Amount']:.2f}")

print(f"Status: {status}, Fraud Probability: {prob:.2%}")

if pred == 1:

print("Action: Flag for review - High-risk transaction detected!")

# Step 6: Explain Flagged Transactions (Feature Importance for Fraud Cases)

if any(predictions == 1):

print("\nFeature Importance for Fraud Detection:")

feature\_importance = pd.DataFrame({

'Feature': X.columns,

'Importance': model.feature\_importances\_

}).sort\_values('Importance', ascending=False)

print(feature\_importance)